ลิสติงที่ 1 (ต่อ)

```
00111:  begin
00112:    fillchar( pck, sizeof( Tregister2NetR ), 0 );
00113:    myName:=TCP1.LocalHostName;
00114:    with pck do
00115:    begin
00116:      msgType:=MSG_SYS_REG; msgLen:=sizeof( Tregister2NetR );
00117:       regType:=rType;      regName:=myName;
00118:      regIp:=TCP1.LocalIP;  regId:=myRegId;
00119:    end;
00120:    VarArrayRedim( txBufferV, sizeof( Tregister2NetR ) );
00121:    p:=VarArrayLock( txBufferV );
00122:    move( pck, p^, sizeof( Tregister2NetR ) );
00123:    VarArrayUnlock( txBufferV );
00124:    TCP2.SendData( txBufferV );
00125:  end;
00126:  procedure TFnetChat.RegisterToNet;
00127:  begin
00128:    CreateRegMesg( REG_LOGIN ); Schedule;
00129:  end;
00130:  procedure TFnetChat.ReleaseFromNet;
00131:  begin
00132:    IF ( TCP2.State = sckConnected ) then
00133:    begin CreateRegMesg( REG_LOGOUT ); Delay( 500 ); end;
00134:  end;
00135:  procedure TFnetChat.SendPackage(pck:T2KBS; pckSize:word; parFlg:boolean);
00136:  var p : pointer; c : integer;
00137:  begin
00138:    if ( parFlg ) then
00139:    begin
00140:      if ( TCP2.state in [ sckConnected, sckConnecting ] ) then
00141:       begin
00142:        VarArrayRedim( txBufferV, pckSize );
00143:        p:=VarArrayLock( txBufferV );  move( pck, p^, pckSize );
00144:        VarArrayUnlock( txBufferV ); TCP2.SendData( txBufferV ); Schedule;
00145:       end;
00146:    end;
00147:    for c:= 0 to MAX_CNT_SCKT-1 do
00148:      if ( FcntSckt[ c ]<>nil ) then FcntSckt[ c ].SendPackage( pck, pckSize  );
00149:  end;
00150:  procedure TFnetChat.SendContent( cntMesg : string );
00151:  var pck : TcontentBlockR; pckP : ^T2KBS;
00152:  begin
00153:    fillchar( pck, sizeof( TcontentBlockR ), 0 ); pckP:=@pck;
00154:    with pck do
00155:    begin
00156:      msgType:=MSG_CONTENT;  msgLen:=sizeof( TcontentBlockR );
00157:      senderName:=myName;
00158:      if ( not confModeFlg ) then GetSelectedClientList( clientList );
00159:      confMode:=confModeFlg; content:=cntMesg;
00160:    end;
00161:    SendPackage( pckP^, sizeof( TcontentBlockR ), true );
00162:  end;
00163:  procedure TFnetChat.ToolButton1Click(Sender: TObject);
00164:  begin
00165:    hostComputer:=InputBox('Enter host computer name', 'name or IP', 'PP166M');
00166:    if ( hostComputer<>'' ) then
00167:    begin
00168:      with tcp2 do
00169:      begin
00170:       RemoteHost:=hostComputer; RemotePort:=7;
00171:       Echo( 'Connecting to host : '+hostComputer );
00172:       Connect( RemoteHost, RemotePort );
00173:      end;
00174:    end;
00175:  end;
00176:  procedure TFnetChat.TCP2Connect(Sender: TObject);
00177:  begin
00178:    DispStatus( 'Connected to '+TCP2.RemoteHost ); RegisterToNet;
00179:  end;
00180:  procedure TFnetChat.TCP2DataArrival(Sender: TObject; bytesTotal: Integer);
00181:  var p : pointer;
00182:  begin
00183:    TCP2.GetData( rxBufferV, VT_ARRAY or VT_UI1, bytesTotal );
00184:    p:=VarArrayLock( rxBufferV );
00185:    move( p^, rxBuffer, bytesTotal-1 );
00186:    VarArrayUnlock( rxBufferV );
00187:    DispatchMessage( bytesTotal-1 );
00188:  end;
00189:
00190:  //- Processing Unit ----------------------
00191:
00192:  procedure TFnetChat.InitVar;
00193:  begin
00194:    with FnetChat do
00195:    begin
00196:      left:=0; top:=0;
00197:      width:=GetSystemMetrics( SM_CXFULLSCREEN )-50;
00198:      height:=GetSystemMetrics( SM_CYFULLSCREEN )-50;
00199:    end;
00200:    myName:=TCP1.LocalHostName; myRegId:=Double( Time );
00201:    Animate1.active:=true;
00202:    confModeFlg:=true;
00203:  end;
00204:  procedure TFnetChat.Echo( content : string );
00205:  begin
00206:    if ( memo1.lines.count>1000 ) then memo1.lines.delete(0);
00207:    memo1.lines.add( content );
00208:  end;
00209:  procedure TFnetChat.wmContentFromClient( var mmsg : TMessage );
00210:  var p : pointer; pck : TcontentBlockR; pckP : ^T2KBS; c : integer;
00211:  begin
00212:    p:=pointer( mmsg.lparam );
00213:    move( p^, pck, sizeof( TcontentBlockR ) );
00214:    ProcessContentFromUser( pck ); pckP:=@pck;
00215:    for c:= 0 to MAX_CNT_SCKT-1 do
00216:      if ( FcntSckt[ c ]<>nil ) then
00217:        if ( FcntSckt[ c ].handle<>mmsg.wparam ) then
00218:          FcntSckt[ c ].SendPackage( pckP^, sizeof( TcontentBlockR ) );
00219:  end;
00220:  procedure TFnetChat.DispatchMessage( bytesTotal : integer );
```

→

ลิสติงที่ 1 (ต่อ)

```
00221:  var curByte,pckSize, msgType : word;
00222:  begin
00223:    curByte:=0;
00224:    repeat
00225:     GetPackageSize( rxBuffer, curByte, msgType, pckSize );
00226:     case msgType of
00227:      MSG_SYS_REG : Unpack_UserConnect2Net( curByte, pckSize );
00228:      MSG_MEMBER_LIST : Unpack_MemberList( curByte, pckSize );
00229:      MSG_CONTENT : Unpack_ContentFromUser( curByte, pckSize );
00230:       else begin
00231:        ShowMessage('Dispatch error [FnetChat], msgType=0x'+IntToHex(msgType,4));
00232:       end;
00233:     end;
00234:     inc( curByte, pckSize+1 );
00235:    until ( curByte >= bytesTotal ) or ( pckSize=0 );
00236:  end;
00237:  procedure TFnetChat.Unpack_UserConnect2Net( curByte, pckSize : word );
00238:  var userReg : Tregister2NetR; listItem : TListItem; idxNo : integer;
00239:  begin
00240:    move( rxBuffer[ curByte ], userReg, pckSize );
00241:    case userReg.regType of
00242:      REG_LOGIN  :
00243:      begin
00244:       with ListView1 do
00245:        begin
00246:         listItem:=Items.Add; listitem.Caption:=userReg.regName;
00247:         listItem.ImageIndex:=0;
00248:        end;
00249:      end;
00250:      REG_LOGOUT :
00251:      begin
00252:       listItem:=ListView1.FindCaption(0, userReg.regName, false, true, true);
00253:       if ( listItem<>nil ) then
00254:        begin
00255:         idxNo:=ListView1.Items.IndexOf( listItem );
00256:         if ( idxNo>-1 ) then ListView1.Items.Delete( idxNo );
00257:        end;
00258:      end;
00259:    end;
00260:  end;
00261:  procedure TFnetChat.Unpack_MemberList( curByte, pckSize : word );
00262:  var mbPck : TnetMemberR; listItem : TlistItem;
00263:  begin
00264:    move( rxBuffer[ curByte ], mbPck, pckSize );
00265:    with ListView1 do
00266:    begin
00267:     listItem:=Items.Add; listitem.Caption:=mbPck.memberName;
00268:     listItem.ImageIndex:=0;
00269:    end;
00270:  end;
00271:  procedure TFnetChat.Unpack_ContentFromUser( curByte, pckSize : word );
00272:  var pck : TcontentBlockR; pckP : ^T2KBS;
00273:  begin
00274:    move( rxBuffer[ curByte ], pck, pckSize ); pckP:=@pck;
00275:    ProcessContentFromUser( pck );
00276:    SendPackage( pckP^, sizeof( TcontentBlockR ), false );
00277:  end;
00278:  procedure TFnetChat.GetSelectedClientList( var cList : T1KBS );
00279:  var listItem : TListItem; cnt : word; clientName : string; c, p : integer;
00280:  begin
00281:    if ( ListView1.SelCount<1 ) then begin confModeFlg:=true; exit; end;
00282:    if ( ListView1.SelCount=1 ) then
00283:    begin
00284:     listItem:=ListView1.Selected; clientName:=listItem.Caption+',';
00285:     cnt:=1; move( cnt, cList, sizeof( word ) );
00286:     move( clientName[1], cList[ sizeof(word) ], length( clientName ) );
00287:    end else
00288:    begin // multi selected
00289:     p:=sizeof( word ); c:=0; cnt:=0;
00290:     repeat
00291:      if ( ListView1.Items[ c ].Selected ) then
00292:       begin
00293:        clientName:=ListView1.Items.Item[ c ].caption+',';
00294:        if ( p+length( clientName )>sizeof( T1KBS ) ) then break;
00295:        move( clientName[1], cList[ p ], length( clientName ) );
00296:        inc( p, length( clientName ) ); inc( cnt );
00297:       end;
00298:       inc( c );
00299:     until ( c>ListView1.Items.Count-1 );
00300:     move( cnt, cList, sizeof( word ) );
00301:    end;
00302:  end;
00303:  procedure TFnetChat.ProcessContentFromUser( pck : TcontentBlockR );
00304:  var clientCnt,p,c:word; clientName:string; separatorP,clientListP:pchar;
00305:  begin
00306:    with pck do
00307:    begin
00308:     if ( confMode ) then
00309:     begin
00310:      GetContent( senderName, content );
00311:     end else
00312:     begin
00313:      p:=sizeof( word ); move( clientList, clientCnt, p );
00314:      clientListP:=@clientList[ p ];
00315:      for c:= 1 to clientCnt do
00316:      begin
00317:       separatorP:=StrPos( clientListP, ',' );
00318:       move( clientListP^, clientName[1], separatorP-clientListP );
00319:       clientName[0]:=chr(separatorP-clientListP);
00320:       if ( clientName=myName ) then
00321:       begin GetContent( senderName, content ); exit; end;
00322:       clientListP:=separatorP+1;
00323:      end;
00324:     end;
00325:    end;
00326:  end;
00327:  procedure TFnetChat.FormCreate(Sender: TObject);
00328:  begin
00329:    InitVar; StartNetChat;
00330:  end;
```

→

ลิสติงที่ 1 (ต่อ)

```
00331:  procedure TFnetChat.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
00332:  begin
00333:    ReleaseFromNet; ReleaseVar; CanClose:=true;
00334:  end;
00335:  procedure TFnetChat.ToolButton17Click(Sender: TObject);
00336:  begin
00337:    if ( MessageDlg( 'Clear all content of Inbox Screen, sure?',
00338:         mtConfirmation, [mbYes, mbNo], 0) = mrYes ) then
00339:      Memo1.lines.Clear;
00340:  end;
00341:  procedure TFnetChat.ToolButton6Click(Sender: TObject);
00342:  begin
00343:    if ( MessageDlg( 'Clear all content of outBox Screen, sure?',
00344:         mtConfirmation, [mbYes, mbNo], 0) = mrYes ) then
00345:      Memo2.lines.Clear;
00346:  end;
00347:  procedure TFnetChat.Memo2KeyDown(Sender: TObject; var Key: Word;
00348:    Shift: TShiftState);
00349:  begin
00350:    if ( key=VK_RETURN ) then SendContent( memo2.lines[ memo2.lines.count-1 ] );
00351:  end;
00352:  procedure TFnetChat.ListView1Click(Sender: TObject);
00353:  var s : string; c : integer;
00354:  begin
00355:    if ( ListView1.SelCount=0 ) then confModeFlg:=true
00356:    else confModeFlg:=false;
00357:    if ( ListView1.selCount>0 ) then
00358:    begin
00359:      s:='User selected : ';
00360:      for c:= 0 to ListView1.Items.count-1 do
00361:        if ( ListView1.Items[ c ].Selected ) then
00362:          s:=s+ListView1.Items.Item[ c ].caption+'. ';
00363:    end else s:='All user selected.';
00364:    DispStatus( s );
00365:  end;
```

myRegId : double; // เก็บรหัส ID ของโปรแกรม ผมกำหนดให้เครื่องแต่ละตัวมี ID ที่ไม่ซ้ำเพื่อที่จะได้ใช้ประโยชน์ในอนาคตได้

myName : string; // เก็บชื่อของเครื่องที่รันโปรแกรม ใช้สำหรับส่งให้สมาชิกภายในระบบรู้จัก

hostComputer : string; // เก็บชื่อเครื่องที่เราทำการติดต่อ (Connect) ด้วย

txBufferV : OLEVariant; // ตัวแปรชนิด OLE Variant ใช้สำหรับระบบ ActiveX Control เป็นหน่วยความจำ Buffer สำหรับส่งข้อมูล

rxBufferV : ใช้เป็นหน่วยความจำ Buffer สำหรับรับข้อมูล ใช้ในส่วน ActiveX Control เท่านั้น

rxBuffer : T64KBS; Buffer สำหรับรับข้อมูลที่ส่งมาจากสมาชิก เป็นหน่วยความจำที่ใช้บันทึกชุดข้อมูลที่จะนำไปตีความ (Dispatch/Unpack)

Listening Unit

รอรับการติดต่อจากสมาชิกใหม่ เมื่อมีสมาชิกใหม่ติดต่อเข้าสู่ระบบแล้ว จะทำการตรวจสอบ Node Connection Unit (uCntSckt.pas) ที่ใช้งานอยู่ ถ้ามี Node ไหนว่างจะทำการมอบหมายให้รับการติดต่อจากสมาชิกใหม่ และจะกลับไปรอการติดต่อจากสมาชิกใหม่อื่น ๆ อีกครั้ง กิจกรรมที่สำคัญในโปรแกรมของส่วนนี้จะประกอบไปด้วย

procedure TFnetChat.TCPListening; เป็นกิจกรรมสำหรับกำหนดให้ TCP1 เปิดบริการ (Service) รอการติดต่อจาก User อื่น ๆ ด้วยการกำหนดหมายเลขพอร์ตสื่อสารให้กับ LocalPort (เท่ากับ 7) แล้วเรียกกิจกรรม TCP1.Listen เพื่อรอการติดต่อ

function TFnetChat.GetFreeConnectionSocket : integer; กิจกรรมสำหรับค้นหา Node Connection Unit ที่ว่าง กิจกรรมนี้ใช้วิธีการค้นหาอย่างง่ายเนื่องจากมีจำนวน source ค่อนข้างน้อย (ไม่เกิน 10) จึงไม่จำเป็นที่ต้องสร้าง Search Engine ที่ซับซ้อน การทำงานของโปรแกรมจะตรวจสอบว่าฟอร์ม FcntSckt (เป็นฟอร์มที่กำหนดไว้เป็นอะเรย์ (Array) สำหรับใช้เป็นส่วน Node Connection Unit) ณ ตำแหน่งที่ระบุได้มีการเรียกใช้หรือยัง ถ้ายังให้ส่งตำแหน่งที่ว่างนี้กลับไป

procedure TFnetChat.ReleaseConnectionSocket(sckNum : integer); กิจกรรมสำหรับยกเลิก Node Connection Unit ที่ใช้งานอยู่ ใช้เมื่อสมาชิกทำการปิดระบบที่ติดต่อ Node Connection Unit จะทำการส่งเมสเสจบอกให้โปรแกรมทำการปิดการให้บริการ ด้วยคำสั่ง Release เพื่อยกเลิกฟอร์มที่ใช้ (ถอนออกจากหน่วยความจำ) และกำหนดให้ชี้ไปที่ nil

procedure TFnetChat.wmIamClose(var mmsg : TMessage); กิจกรรมสำหรับดำเนินการปิดการใช้บริการ โดยจะยกเลิกไอคอนที่ใช้แทนสมาชิก (ที่ยกเลิก) ออกจากระบบ ด้วยคำสั่ง FindCaption เพื่อค้นหาชื่อที่ใช้ลงทะเบียนไว้ (pck.regName) แล้วลบออก ก่อนเรียกกิจกรรม ReleaseConnectionSocket ดังนี้

```
listItem:=ListView1.FindCaption
( 0, pck.regName, false, true, true );
if ( listItem<>nil ) then
begin
      idxNo:=ListView1.Items.IndexOf( listItem );
      if ( idxNo>-1 ) then ListView1.Items
      .Delete( idxNo );
end;
ReleaseConnectionSocket( Integer( mmsg.wparam ) );
```

procedure TFnetChat.wmUserConnect(var mmsg : TMessage); กิจกรรมสำหรับรับแจ้งการติดต่อจากสมาชิกใหม่ กิจกรรมนี้จะทำงานเมื่อสมาชิกใหม่ทำการติดต่อเข้ามาในระบบเรียบร้อย พร้อมส่งชุดข้อมูลลงทะเบียน เพื่อขอลงทะเบียนในระบบ กิจกรรมจะตรวจสอบชนิดการลงทะเบียน ถ้าเป็น REG_LOGIN จะเพิ่มชื่อของสมาชิกไปแสดงไว้ใน ListView1 และจะส่งรายชื่อสมาชิกในระบบที่ติดต่ออยู่ในขณะนั้นให้กับสมาชิกด้วยกิจกรรม IntroductMember2User พร้อมกับแนะนำสมาชิกรายนี้ให้กับกลุ่ม ด้วยกิจกรรม UserConnect2Net

procedure TFnetChat.IntroductMember2User(clnt : HWnd); กิจกรรมสำหรับส่งรายชื่อสมาชิกในระบบทั้งหมดให้กับสมาชิกใหม่ โดยส่งแอดเดรสของ ListView1 ผ่านกระบวนการส่งเมสเสจให้กับ Node Connection Unit เพื่อค้นหารายชื่อสมาชิกในระบบ แล้วส่งให้สมาชิกใหม่

procedure TFnetChat.UserConnect2Net(clnt : HWnd; pck : Tregister2NetR); กิจกรรมสำหรับแนะนำสมาชิกใหม่ที่พึ่งติดต่อเข้าระบบให้สมาชิกทั้งหมดรู้จัก โดยการส่งชุดข้อมูล pck ให้กับสมาชิกภายในกลุ่ม (ส่งให้เครื่องแม่ด้วยถ้ามีการติดต่ออยู่) ดังนี้

```
// ส่งให้สมาชิกภายในกลุ่มเดียวกัน
for c:= 0 to MAX_CNT_SCKT-1 do
 if ( FcntSckt[ c ]<>nil ) then
  if ( FcntSckt[ c ].handle<>clnt ) then
   FcntSckt[ c ].SendPackage( pckP^, sizeof
   ( Tregister2NetR ) );
// ส่งให้เครื่องแม่ที่ติดต่ออยู่
if ( TCP2.state in [ sckConnected, sckConnecting ] )
then
begin
 VarArrayRedim( txBufferV, sizeof( Tregister2NetR ) );
 p:=VarArrayLock( txBufferV );
 move( pck, p^, sizeof( Tregister2NetR ) );
 VarArrayUnlock( txBufferV );
 TCP2.SendData( txBufferV );
 Schedule;
end;
```

procedure TFnetChat.TCP1ConnectionRequest(Sender: TObject; requestID: Integer); กิจกรรมสำหรับรับการติดต่อจากเครื่องอื่นที่ต้องการเป็นสมาชิกของกลุ่มนี้ โปรแกรมจะค้นหา Node Connection ที่ว่างด้วยกิจกรรม GetFreeConnectionSocket และทำการสร้าง Node Connection Form เพื่อให้รับการติดต่อจากสมาชิกใหม่ด้วยกิจกรรม Accept ดังนี้

```
newSckt:=GetFreeConnectionSocket;
if ( newSckt>-1 ) then
begin
 if ( FcntSckt[ newSckt ]=nil ) then
  Application.CreateForm( TFcntSckt,
  FcntSckt[ newSckt ] );
 FcntSckt[ newSckt ].Show; Schedule;
 FcntSckt[ newSckt ].Tcp1.Accept( requestId );
 with FcntSckt[ newSckt ].myInfo do
 begin
  netChatHdl:=FnetChat.Handle;   cntScktNo:=newSckt;
  remoteIP:=FcntSckt[ newSckt ].TCP1.RemoteHostIP;
  FcntSckt[ newSckt ].caption:=remoteIP; Schedule;
  FcntSckt[ newSckt ].WindowState:=wsMinimized;
  Schedule;
 end;
end;
```

Connection Unit

ติดต่อไปยังระบบอื่น ๆ เพื่อขอเข้ากลุ่มเป็นสมาชิกในกลุ่มนั้น โดยเมื่อติดต่อสำเร็จจะทำการส่ง Register Package ไปลงทะเบียนเป็นสมาชิกใหม่ พร้อมรับรายชื่อสมาชิกทั้งหมดในระบบ เพื่อใช้ระบุชื่อเมื่อต้องการส่งข้อมูลไปให้ กิจกรรมที่สำคัญในโปรแกรมของส่วนนี้จะประกอบไปด้วย

procedure TFnetChat.CreateRegMesg(rType : word); กิจกรรมสำหรับสร้างชุดข้อมูลลงทะเบียน และส่งข้อมูลไปลงทะเบียน ชุดข้อมูลสำหรับลงทะเบียนจะประกอบไปด้วย msgType ระบุชนิดของชุดข้อมูล, regType ระบุชนิดการขอลงทะเบียน ถ้าเป็น REG_LOGIN คือขอลงทะเบียนเป็นสมาชิกใหม่ และ REG_LOGOUT เมื่อต้องการเลิกการติดต่อ, regName ระบุชื่อเครื่องของสมาชิกใหม่

procedure TFnetChat.RegisterToNet; กิจกรรมสำหรับลงทะเบียนเป็นสมาชิกใหม่ โดยผ่านค่า REG_LOGIN ให้กับกิจกรรม CreateRegMesg เพื่อส่งชุดข้อมูลไปลงทะเบียน

procedure TFnetChat.ReleaseFromNet; กิจกรรมสำหรับยกเลิกการติดต่อกับระบบ ผ่านค่า REG_LOGOUT ให้กับกิจกรรม CreateRegMesg เพื่อทำการ Log out ออกจากระบบ

procedure TFnetChat.SendPackage(pck : T2KBS; pckSize : word; parFlg : boolean); กิจกรรมสำหรับส่งชุดข้อมูลให้กับเครื่องที่ติดต่ออยู่ โดยตรวจสอบดูว่าส่วน Connection Unit (TCP2) ติดต่ออยู่กับเครื่องแม่หรือไม่ ถ้าติดต่ออยู่ให้ส่งข้อมูลไปให้ และเช็ค Node Connection Unit ว่ามีเครื่องลูกติดต่อหรือไม่ ถ้ามีส่งชุดข้อมูลไปให้เช่นกัน

procedure TFnetChat.SendContent(cntMesg : string); กิจกรรมส่งข้อความให้กับระบบที่ติดต่ออยู่ กิจกรรมนี้จะเรียกใช้กิจกรรม SendPackage เพื่อส่งชุดข้อมูลให้กับสมาชิกในกลุ่มทั้งหมด เราสามารถสร้างกิจกรรมเพื่อส่งชุดข้อมูลอื่น ๆ เช่น ส่งแฟ้มข้อมูล ส่งรูปภาพ ฯลฯ ได้อีกมาก โดยใช้หลักการเดียวกับกิจกรรม SendContent

procedure TFnetChat.ToolButton1Click(Sender: TObject); กิจกรรมชนิด Event ของ ToolBar ปุ่ม 'Connect' สำหรับระบุชื่อเครื่องที่ต้องการติดต่อด้วย InputBox และส่งให้ Connection Unit (TCP2) พยายามติดต่อไปหา

procedure TFnetChat.TCP2Connect(Sender: TObject); กิจกรรมชนิด Event ของ TCP2 เป็นกิจกรรมที่เกิดขึ้นหลังจากที่เครื่องแม่รับการติดต่อจาก TCP2 จึงนำเอากิจกรรม RegisterToNet มาใส่เพื่อให้ส่งชุดข้อมูลสำหรับลงทะเบียนแบบอัตโนมัติทันทีที่สามารถติดต่อได้

procedure TFnetChat.TCP2DataArrival(Sender: TObject; bytesTotal: Integer); กิจกรรมชนิด Event ของ TCP2 เป็นกิจกรรมใช้สำหรับรับข้อมูลจาก TCP ActiveX Control แล้วส่งให้กับกิจกรรม DispatchMessage เพื่อตีความ

Processing Unit

ส่วนควบคุมการแสดงผล การตีความ การเชื่อมต่อฟอร์มรองเข้ากับฟอร์มหลัก และควบคุมคอนโทรลของหน้าต่างหลัก กิจกรรมที่สำคัญในโปรแกรมของส่วนนี้จะประกอบไปด้วย

procedure TFnetChat.InitVar; กิจกรรมสำหรับ Initializing ระบบ เพื่อเตรียมรันโปรแกรม และปรับขนาดหน้าต่างแสดงผลให้มีขนาดใหญ่พอสมควร

procedure TFnetChat.Echo(content : string); กิจกรรมแสดงผลข้อความออก Memo1

procedure TFnetChat.wmContentFromClient(var mmsg : TMessage); กิจกรรมชนิด Event Message สำหรับรับชุดข้อมูลชนิดข้อความที่ส่งมาจาก Node Connection Unit เพื่อแสดงผล โดยส่งให้กิจกรรม ProcessContentFromUser และส่งให้สมาชิกอื่นในกลุ่มเดียวกัน

procedure TFnetChat.DispatchMessage(bytesTotal : integer); กิจกรรมตีความหมายชุดข้อมูลที่รับจาก TCP ActiveX Control

ในกิจกรรมชุดนี้รองรับการตีความหมายของชนิดข้อมูล 3 แบบ คือ MSG_SYS_REG (ชุดข้อมูลสำหรับลงทะเบียน) MSG_MEMBER_LIST (ชุดรายชื่อสมาชิกในระบบ) และ MSG_CONTENT (ชุดข้อความที่ส่ง)

procedure TFnetChat.Unpack_UserConnect2Net(curByte, pckSize : word); กิจกรรมตีความชุดข้อมูลลงทะเบียน ถ้าเป็นชุดข้อมูลชนิด REG_LOGIN จะทำการเพิ่มชื่อสมาชิกเข้าสู่ ListView1 และถ้าเป็น REG_LOGOUT จะทำการลบชื่อสมาชิกจาก ListView1

procedure TFnetChat.Unpack_MemberList(curByte, pckSize : word); กิจกรรมสำหรับตีความรายชื่อสมาชิกที่ส่งให้ เพื่อเพิ่มเข้าสู่ ListView1

procedure TFnetChat.Unpack_ContentFromUser(curByte, pckSize : word); กิจกรรมสำหรับตีความข้อความที่รับมาพร้อมกับส่งให้สมาชิกอื่นในระบบ

procedure TFnetChat.GetSelectedClientList(var cList : T1KBS); กิจกรรมสำหรับเลือกรายชื่อสมาชิกใน ListView1 เพื่อทำการติดต่อกับสมาชิกบางคนเท่านั้น รายชื่อสมาชิกที่เลือกจะถูกบรรจุในตัวแปร cList โดยใช้หน่วยความจำตำแหน่งแรก (มีขนาดเท่ากับ Word) เก็บจำนวนรายชื่อที่เลือก และรายชื่อทั้งหมดที่บรรจุใน cList จะถูกคั่นด้วย ',' (เครื่องหมาย Comma)

procedure TFnetChat.ProcessContentFromUser(pck : TcontentBlockR); กิจกรรมสำหรับแสดงผลชุดข้อมูลชนิด Content โดยตรวจสอบ Mode การส่งถ้า ConfMode = TRUE หมายถึงสมาชิกทุกคนที่ติดต่ออยู่ในระบบมีสิทธิเข้าถึงข้อมูลที่ส่งได้ แต่ถ้าเป็น FALSE จะทำการตรวจสอบรายชื่อสมาชิกที่มีสิทธิที่ส่งมาพร้อมกับข้อความ ถ้ามีรายชื่อระบุ clientName ตรงกับ myName (เครื่องเราที่ใช้อยู่) ก็สามารถมีสิทธิเข้าถึงข้อมูลได้ (จะเรียกกิจกรรม GetContent เพื่อแสดงข้อความ)

procedure TFnetChat.Memo2KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState); กิจกรรมชนิด Event สำหรับตรวจสอบการพิมพ์ใน Memo2 ถ้ามีการกดปุ่ม [Enter] ก็จะนำเอาข้อมูลในบรรทัดใหม่ส่งให้กับสมาชิกในระบบ

ขั้นตอนการนำไปใช้งาน

Register ActiveX Control

การนำโปรแกรม NetChat ไปใช้งานมีขั้นตอนการเตรียมโปรแกรมเล็กน้อย เนื่องจากระบบ TCP/IP Object ที่ใช้เป็น ActiveX Control จึงต้องมีการเตรียมการดังนี้

- คัดลอกไฟล์ WINSOCK.OCX, NMOCOD.DLL, NMORENU.DLL, NMSCKN.DLL และ OLEPRO32.DLL จาก Folder \Windows\System ไปยังเครื่องที่ต้องการรันโปรแกรมที่ Folder เดียวกัน
- ใช้โปรแกรม REGSVR32.EXE (ของไมโครซอฟต์วินโดวส์) ทำการ Register ActiveX Control ตามลำดับต่อไปนี้

1. REGSVR32 /S NMOCOD.DLL
2. REGSVR32 /S WINSOCK.OCX

ติดตั้ง TCP/IP Protocol

ระบบ NetChat ใช้ TCP/IP Protocol สื่อสารกันระหว่างเครื่อง ดังนั้นเครื่องที่ต้องการนำโปรแกรม NetChat ไปใช้งานจึงจำเป็นต้องติดตั้ง TCP/IP Protocol เสียก่อนดังนี้

- กดปุ่ม [เริ่ม Start] → เลือกเมนู Settings → เลือกเมนู Control Panel
- ดับเบิลคลิกที่ไอคอน Network
- กดปุ่ม [Add] → เลือกรายการ Protocol จาก ListBox → กดปุ่ม [Add] ในหน้าต่าง 'Select Network Components Type'
- เลือกรายการ 'Microsoft' ใน Manufacturers List Box แล้วเลือกรายการ 'TCP/IP' ใน Network Protocols List Box → กดปุ่ม [OK]
- วินโดวส์จะทำการติดตั้ง TCP/IP Protocol โดยถามหาแผ่นโปรแกรมติดตั้ง ให้ทำตามคำสั่งของวินโดวส์

● เมื่อติดตั้งเสร็จแล้วจะกลับมาที่หน้าต่าง Network อีกครั้ง จะเห็นรายการ TCP/IP ใน List Box ทำการ Reboot ตามขั้นตอนของวินโดว์อีกครั้ง

ทดลองรันโปรแกรม

หลังจากติดตั้ง TCP/IP และลงโปรแกรม NetChat (รวมทั้ง Register ActiveX Control) เรียบร้อย สามารถทดลองรันโปรแกรมได้เลยโดยคลิกที่โปรแกรม NETChat.EXE แล้วกดปุ่ม [Connect] ใส่ชื่อเครื่องคอมพิวเตอร์ของเครื่องที่ต้องการติดต่อ โปรแกรมจะทำการติดต่อไปยังเครื่องที่ระบุทันที

ปัญหา และบัก (Bug)

ถ้าหากโปรแกรมแจ้งความผิดพลาดเกี่ยวกับ ActiveX Control คุณอาจจะมีปัญหาเกี่ยวกับการ Register ActiveX Control ขอให้ทบทวนขั้นตอนการติดตั้งอีกครั้ง

ถ้าขั้นตอนการติดตั้งต่าง ๆ เรียบร้อย แต่โปรแกรมไม่สามารถติดต่อกันได้ อาจมีปัญหาเนื่องจากเครื่องคอมพิวเตอร์ไม่สามารถมองเห็นกันได้ คุณสามารถทดสอบได้โดย ออกไปที่ DOS Prompt พิมพ์คำสั่ง WINIPCFG.EXE ให้อ่านค่า IP Address ที่แสดงอยู่ในรูปที่ 2 คือ 129.103.99.1 ในหน้าต่าง IP Configuration แล้วไปที่เครื่องที่ต้องการติดต่อ ให้ลองพิมพ์คำสั่ง 'C:\>WINDOWS\PING 129.103.99.1'

ถ้าสามารถติดต่อได้โปรแกรมจะแสดงข้อความ "Reply from 129.103.99.1 : bytes = 32 time...." แสดงว่าระบบ TCP/IP ของคุณสมบูรณ์ คุณอาจต้องตรวจสอบการใช้งานอีกครั้ง

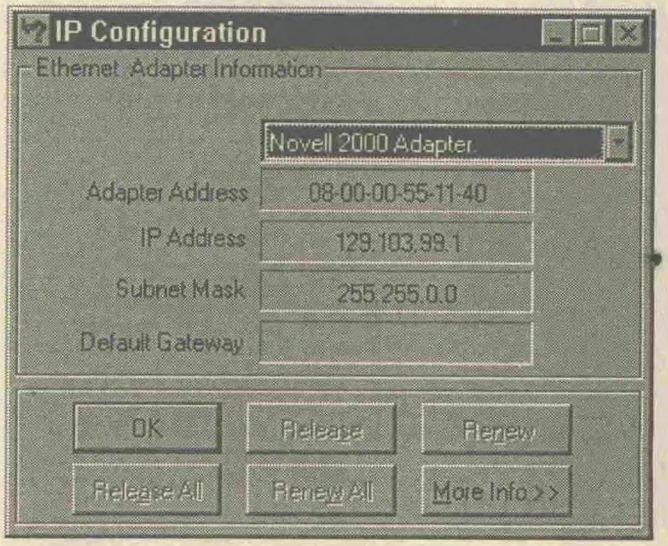
ถ้าไม่สามารถติดต่อได้ คุณอาจต้องติดต่อ Network Administrator หรือผู้ดูแลระบบ Network ให้ช่วยแก้ปัญหา Network

สำหรับในกรณีที่ระบบ Network สมบูรณ์ทุกอย่าง แต่โปรแกรมไม่สามารถติดต่อได้ ให้ลองกำหนดค่า IP Address แบบตายตัว ทำได้โดย

- เลือกไอคอน Network ใน Control Panel
- คลิกที่รายการ TCP/IP ใน List Box → กดปุ่ม [Properties]
- ใน IP Address Page ให้เลือก "Specifies an IP Adress" ให้กำหนด

IP Address เป็น '129.100.100.XXX' โดย XXX มีค่าระหว่าง 0-200 แล้วกดปุ่ม [OK] เพื่อ Reboot เครื่องใหม่อีกครั้ง

ตอนนี้ เครื่องที่ติดตั้งโปรแกรม NetChat ควรที่จะสามารถทำงานเป็นปกติได้แล้ว ปัญหาอีกอย่างที่ค่อนข้างสำคัญในการพัฒนาโปรแกรมนี้ต่อเพื่อนำไปใช้งานจริงคือ ปัญหาความผิดพลาดในโปรแกรม (บัก : Bug) ซึ่งมักจะพบบ่อย ๆ ในโปรแกรมที่พัฒนาขึ้นเอง



รูปที่ 2

ในการพัฒนาโปรแกรมนี้ ผมตั้งใจจะเขียนให้เป็นแนวทางในการใช้งาน TCP/IP เพื่อการสื่อสารกันแพร่หลายมากขึ้น และประจวบกับมีเวลาในการทดสอบโปรแกรมค่อนข้างน้อย จึงไม่ได้เขียนโปรแกรมในส่วนตรวจสอบความผิดพลาดให้ ซึ่งคงต้องเป็นหน้าที่ของผู้สนใจจริง ๆ นำไปค้นคว้าเพิ่มเติมต่อเอง สำหรับผู้ที่สนใจซอร์สโค้ดโปรแกรมตัวเต็มของโปรแกรม NetChat เชิญใช้บริการซอฟต์เมล์ประจำฉบับได้เลยครับ

ฉบับนี้คงต้องกล่าวคำอำลาเพียงเท่านี้ ฉบับหน้าพบกันใหม่สวัสดีครับ